Iterative Analysis of Pages in Document Collections for Efficient User Interaction

Joseph CHAZALON, Bertrand COÜASNON
INSA Rennes – UEB
UMR IRISA
Rennes, France
{*joseph.chazalon,bertrand.couasnon*}*@irisa.fr*

Aurélie LEMAITRE
Université de Rennes 2 – UEB
UMR IRISA
Rennes, France
aurelie.lemaitre@irisa.fr

***Abstract*—The analysis of sets of degraded documents, like historical ones, is error-prone and requires human help to achieve acceptable quality levels. However, human interaction raises 3 main issues when processing important amounts of pages: none of the user or the system should wait for work; information provided by a human operator should not be restricted to local isolated corrections, but rather produce durable changes in the system; the ability to interact with a human operator should not increase the complexity of document models nor duplicate them between analysis and human interaction processes. To solve those issues, we propose an iterative approach, based on a special mechanism called visual memory, to reintegrate external information during page analysis. So as to demonstrate the interest for existing systems, we explain how we adapted a (rule-based) page analysis tool to enable, in this iterative approach, a delayed interaction with a human operator based on an adaptation of error recovery principles for compilers and the well-known exception handling mechanism. We validated our iterative approach on sales registers from the 18th century.**

***Keywords*-document analysis; degraded documents; document sets; iterative analysis; user interaction;**

I. INTRODUCTION

Degraded documents (like historical ones) prevent automated analysis from achieving sufficient quality level, and for many tasks, human intervention is necessary to bridge the semantic gap between machine and man. We identified 3 requirements for this kind of interaction.

As sets of documents we process are often made of tens of thousands of pages, *interacting with a human must not make the human operator wait for work nor block the system on each uncertainty which may arise (Req. 1: Viability).*

Structure and content recognition are difficult in those documents, and existing systems use sophisticated document models to extract structured data. Then, *interaction must not increase model complexity with synchronization considerations, nor fragment document knowledge (Req. 2: Simplicity),* in particular between analysis and human interface processes to model-check answers supplied by a human.

When ran, the analysis system we use produces lots of errors which are often similar or linked, as document sets are organized in *collections* presenting stabilities in page layouts and contents. For example, in neighbor pages of the tabular document in Fig. 1, words written in a particular

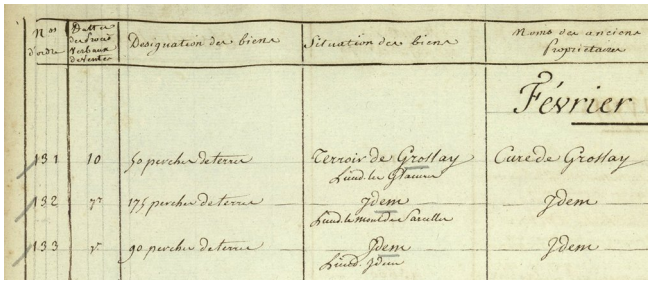

Figure 1. Excerpt from a page of a tabular document (18^{th} century).

column belong to the same lexicon. Error corrections are also often similar or linked, like when, in a sequence of numbers, correcting a number provides some contextual knowledge for the others. Therefore, the *interaction should be* durable*, in the sense of [1], and permit the propagation of corrections, the reprocessing of previous results, and even system improvement, instead of only local and isolated corrections (Req. 3: Durability).*

We first consider, in Sec. II, how existing approaches of interactive analysis comply with those requirements. Sec. III presents our contribution to satisfy those needs: an iterative analysis approach for which we exhibit the minimal architecture required. We focus on how to turn existing isolated page analyzers into interactive ones suitable for the processing of document collections, using a visual memory mechanism. Sec. IV details the modifications we made to an existing (rule-based) page analysis tool to enable, in this iterative framework, an *asynchronous machine-initiated* interaction with a human operator. This implementation relies on broadly available features easily adaptable to existing page analyzers, and was tested on production tasks.

II. RELATED WORK

The simplest way to improve analysis results with human interaction is post-processing error correction. PerfectDoc [2] is a post-processing tool which enables bulk correction of document structure and character recognition. While the human operator does not wait for tasks, the system does not validate human information nor it uses it to correct previous

interpretations. Therefore, we consider the human and the system should *cooperate during* the analysis phase, not *after*.

Another interesting approach is DocMining [3], a very general framework for document processing where document processing units are successively invoked by a scheduling module following a predefined scenario. A persistent structure is associated to each document where each processing unit consumes and produces elements. DocMining answers several parts of our problem: the *central storage* module enables a delayed interaction between components; the *persistent structure associated to each image* permits a simple information fusion; and *human interaction can be easily be implemented* as a processing units. However, as the persistent structure stores the state of the analysis in this data-driven approach, there is no reprocessing of each page. It limits human interaction to a local substitution to automated processes, and forbids advanced *cooperative* behaviors like model validation, correction of previous analysis stages, or guiding the analysis. Therefore an *iterative analysis* where pages are reprocessed with new external information is also necessary.

The smartFIX system [4] goes a step further in that direction: not only it uses a central repository to enable a delayed interaction, but is also proposes iterative recognition, optimization and control of documents using explicit model definitions: when human operators correct results, they are automatically checked against those models. However, as analysis modules process documents independently, this approach may be somehow limited in the case of collections of old documents. The lack of some *strategy at a collection level* could prevent it from making use of collection properties, like local word repetitions, by grouping similar problems before validation to increase human operator efficiency. Furthermore, as few details are given about information reintegration in analysis processes, it is hard to know whether human interaction could be more elaborated than the labeling of suspicious fields.

While existing approaches do not comply with all the requirements for a *viable, simple* and *durable* interaction when processing document collections, they provide interesting insights. A central database can store information about pages to transmit between processes and enable an asynchronous interaction which avoids them to wait (Req. 1). To remain maintainable, page analyzers have to carry the whole document model and reprocess each page to reintegrate or validate external information. The human interface can then be homogeneously integrated as a page analyzer, without duplicating the document model (Req. 2). This iterative analysis fully makes use of human information, as an efficient interaction relies on the ability to recompute previous results (Req. 3). Furthermore, a practical system also needs: i) a strategy module to coordinate the processes; and ii) a mean to reintegrate external information in page analyzers.

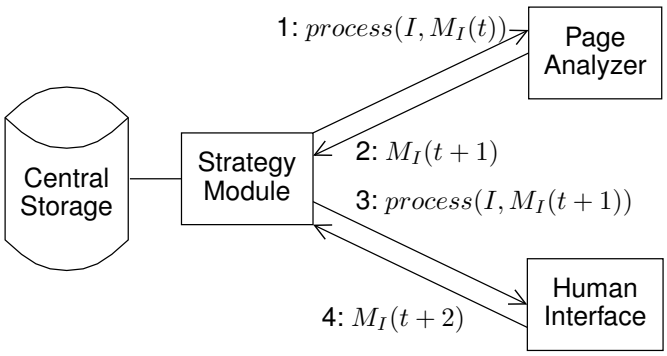

Figure 2. A single step (repeated as many times as needed during the *iterative analysis*) of a simple dialogue strategy between a page analyzer and a human interface, showing the implied components of the architecture and the *visual memory* M_I associated to the image I.

III. Iterative Analysis of Document Pages

We detail here the necessary components (visible in Fig. 2) we identified, then we explain their iterative behavior from a page analyzer's point of view. Finally, we present the interaction models that this iterative approach enables.

A. Required Components and Features

1) Strategy Module: The strategy module is responsible for: i) coordinating the processes; and ii) providing them with appropriate data and gather their results. It acts both as a scheduler following a particular strategy, and a mediator between processes backed by a central database. The behavior of the strategy module should ensure that each page-level process is invoked with enough data to progress through each of its iterative invocations during the whole analysis. Its abstraction level also enables, while we will not present those elements, to: i) make use of collection context; ii) increase human interaction efficiency: for example, by grouping similar questions as we proposed in [5]; and iii) improve the system.

2) Visual Memory: In order to reintegrate external information during the analysis of pages, we use a *visual memory*. It is the support of information exchange and usage between processes. To each page, we associate a persistent dataset transmitted to each process when invoked by the strategy module. Each process can change its content. Every data it contains has a shape and a position in its image's referential (thus the *"visual"*). We note $M_I(t)$ the visual memory related to the image I at time t.

3) Central Database: The central database stores the associated visual memories for each page in the whole collection. It enables: i) an asynchronous interaction between processes; and ii) an improvement of human interaction efficiency using properties of document collections, like by grouping questions about similar fields to speed up their manual correction. It provides a *collection-centric view* of the information stored which can progressively grow.

4) Page Analyzer: The page analyzer uses a document model to try to extract relevant information from an image. In an iterative analysis scheme, it is invoked with a image reference I and the visual memory $M_I(t)$, and returns $M_I(t+1)$ to the strategy module when done. Each iteration *reprocesses the whole image* and enables: i) a validation against the document model (without duplicating it) of external information; and ii) the production of new structured results, based on (integrating) external information.

5) Human Interface: In this presentation, we consider the human interface as a page analyzer. It shows an image and associated memory to the human operator who then proposes new information. In a more general approach, it would have more important links with the strategy module, and would consider elements from different pages.

B. Iterative Analysis Behavior

1) Global Information Flow: Fig. 2 shows a single step (repeated as long as needed) of information exchange (for a given page) between a page analyzer and a user interface in a simple strategy which triggers each component one after another for each page. The communication with the central database is omitted for brevity. At time t, the strategy modules first invokes the page analyzer on the image I, with the memory $M_I(t)$. The strategy module gathers $M_I(t+1)$ after the analysis, and transfers it to the human interface, which, in turn, produces $M_I(t+2)$ and sends it back.

2) External Information Fusion with Visual Memory: Information fusion between the current image I and external information, M_I, can be possible by: i) placing all data in the image referential, with a shape and a position; ii) ensuring that data, whenever it is extracted from the image or external, is available at any moment during the analysis; iii) providing equivalent access operations to each kind of data (read, create, delete at least).

We recompute M_I at the end of each step of this iterative analysis.

C. Available Interaction Models for Page Analyzers

The iterative analysis we propose enables two complementary types of efficient user interaction for page analyzers.

1) Directed Interaction: Like for the *machine-initiated* concept of [1], data to be analyzed by the human operator is determined automatically by an algorithm. We use the term *directed* to indicate the underlying question/answer semantic. It requires the automated system to detect errors and explicitly ask for resolutions. The human operator has no choice on the information to provide, and the system can more easily identify which of its subparts is to be corrected.

2) Spontaneous Interaction: This model is a bit different from the *human-initiated* concept of [1]: the human operator is free to provide or alter any information he wants for a given page, but it may not produce changes in the automated results, as the system may not validate or even consider those elements. Even if spontaneous interaction is necessarily limited to several information types by the system, the difficulty is, for the operator, to provide useful elements, and for the system to use them. On the other side, no automated error detection is required. An example of such interaction is, with the document of Fig. 1, when a human operator notices that the system did not locate a number, and decides to provide its position. The analyzer could then use this new information and integrate it in the structured result.

Interaction at the Strategy Level: As this paper focuses on how to adapt a page analyzer, we do not detail interaction capabilities of the strategy module, inside which: i) interaction can be made more efficient using machine learning, or grouping similar requests as we proposed in [5]; and ii) a "director" user could choose the processes to invoke in a manual scenario, like for the *user-driven* mode of [6].

IV. Implementation of a Page Analyzer Based on Error Detection, Correction and Recovery

We present here how we turned a rule-based page analyzer, based on the DMOS-P approach [7], into an iterative process ready for a efficient *directed* interaction.

A. DMOS-P Framework and Language

DMOS-P [7] is a concept-driven grammatical document analysis method. It uses a bi-dimensional extension of Definite Clause Grammars, called *Enhanced Position Formalism* (EPF), to describe pages. The following example illustrates its syntax.

```
A ::= AT(top) && B. % clause 1, tried 1st
A ::= AT(bottom) && C. % tried if 1 fails
```

To recognize `A`, we try, at the `top` of the image, to recognize `B`, and if it fails, we try, at the `bottom`, to recognize `C`. Like attribute grammars, rules have input and output parameters indicated by "+" and "–" signs, as illustrated by:

```
recognizeNumber(+NumPos, -Value) ::=
   callClassifier(+NumPos, -Value).
```

B. Communication With Other Components

1) Visual Memory Data Structures: To enable the communication with the human, the visual memory carries (and locates in the image referential) *questions* and *answers*, which are represented with the following containers

$$Q(Text, AnswerArea, DataType)$$
$$A(Data)$$

$Text$ gives information about the problem and $DataType$ indicates the expected type for the content of the answer, which is stored in its $Data$ attribute. $AnswerArea$ indicates the area inside which the answer can be located.

2) How Questions Are Answered: Answering the questions is done outside of the page analyzer, in the human-machine interface. At time t, it loads the image I, the associated visual memory $M_I(t)$, and displays the questions. The human operator answers the questions, which are

deleted, and locates each answer *inside* its acceptation zone $AnswerArea$. At the end, $M_I(t+1)$ contains only current and previous answers, kept for later use.

3) Challenges: The implementation has to: i) detect errors and ask questions; ii) ensure that answers are used by the analyzer to make progress instead of asking the same questions forever; and iii) make as much progress as possible in independent parts of the analysis if a problem arises.

C. DMOS-P Extensions for Directed Interaction

Three new *EPF* operators are provided to add the required semantics in page descriptions. Their implementation is an adaptation of error recovery principles for compilers. In logical and functional languages, higher order and continuation easily enable such approach, and in imperative ones, exceptions (and annotations) can be used.

1) Asking Questions (Error Detection): Like one raises an exception, we indicate a problem was detected and ask for external information with the operator

```
raiseQuestion(+Text, +Zone, +DataType)
```

When called, it 1) adds a question in the visual memory M_I, with shape and position defined by `+Zone` to locate the issue; and 2) continues the analysis just after the latest invocation of `catchQuestion`. The $AnswerArea$ attribute of the question is automatically defined as the search position during the last invocation of `getAnswerOrTry`.

2) Using Answer (Error Correction): Like with a *try* keyword in many languages, we can identify the part of the analysis impacted by a given problem with the operator

```
getAnswerOrTry(+DataType, -Result, +Rule)
```

where `+DataType` indicates the type of the content of acceptable answers, and `+Rule` is a rule which has a unique output parameter whose type is `+DataType`. Therefore, the rule and the answer can provide elements of the same type. When called, it looks at the current search position for an answer $A(Data)$ in M_I where the type of $Data$ is `+DataType`. If it exists, the value of `-Result` is $Data$. Otherwise, it invokes the rule `+Rule` and the value of the output parameter of `+Rule` is used as value for `-Result`. To enable interaction on structural elements, a special case returns the zone of the answer if the content of the answer as no value (meaning it is just a marker).

3) Continuing the Analysis (Error Recovery): Like with a *catch* keyword, we indicate an upper bound in an analysis branch where analysis can be safely continued when a given problem arises with the operator

```
catchQuestion(+Rule)
```

where `+Rule` is a rule which may have any parameter. When called, it invokes the rule `+Rule` and catches any interaction request raised with `raiseQuestion` in that rule. If no question is raised, the output parameters of `+Rule` are well defined, otherwise they are left uninstantiated.

D. Usage Example

This example aims at locating and recognizing the numbers contained in the leftmost column of the document shown in Fig. 1, using the trivial dialogue strategy of Fig. 2.

1) Description without Interaction: A simple non-interactive description first locates the left column, and inside this column, locates and recognizes each number. For clarity, we only detail `start` and `readAllNumbers`.

```
start() ::= AT(allPage) &&
   locateLeftCol(-ColPos) &&
   AT(+ColPos) &&
   readAllNumbers().

readAllNumbers() ::=
   locateNumber(-NumPos) &&
   recognizeNumber(+NumPos, -Value) &&
   % use Value...
   AT(under) &&
   % loop until no more numbers...
```

2) Description with Directed Interaction: Useful memory elements are `lCol` (of type `colT`) which indicates the left column position, and `num(Value)` (of type `numT`) which associates a value to a number located in the memory. Modified elements are indicated in **bold**.

```
start() ::= AT(allPage) &&
   getAnswerOrTry(colT, -ColPos,
      locateLeftCol2(-ColPos)) &&
   AT(+ColPos) &&
   readAllNumbers().

readAllNumbers() ::=
   locateNumber(-NumPos) &&
   catchQuestion(
      getAnswerOrTry(
         numT, -Value,
         recognizeNumber2(+NumPos,-Value)
      ))
   % use Value (may be uninstantiated)
   AT(under) &&
   % we still can read other numbers...

locateLeftCol2(-ColPos) ::=
   locateLeftCol(-ColPos).
% If automated version fails, ask.
locateLeftCol2(-ColPos) ::=
   raiseQuestion("Where is the column?",
         allPage, colT).

recognizeNumber2(+NumPos,-Value) ::=
   recognizeNumber(+NumPos,-Value).
recognizeNumber2(+NumPos,-Value) ::=
   raiseQuestion("What is this number?",
         +NumPos, numT).
```

Furthermore, the invocation of the `start` rule is automatically changed to: i) read $M_I(t)$; ii) invoke `start` and catch any question; and iii) write $M_I(t+1)$.

3) Interaction Dialogue: For an image I where the left column cannot be automatically located, and where 2 numbers over 4 are not recognized, a strategy with a simple dialogue between the page analyzer and the human interface (like in Fig. 2) produces the following behavior:

At t_0, the analysis starts.

$$M_I(t_0) = \emptyset$$

From t_0 to t_1, the page analyzer tries to locate the left column and fails: it asks a question.

$$M_I(t_1) = \{Pa \rightarrow Q(\text{“Where is the column?”}, Pa, colT)\}$$

where Pa is a zone covering the whole page area.

From t_1 to t_2, the human operator answers the question and provides the location of the column.

$$M_I(t_2) = \{Ca \rightarrow A(lCol)\}$$

where Ca is the zone where the column is located.

From t_2 to t_3, the page analyzer try to locate the left column and uses the answer. It starts locating and reading numbers, but fails to read two of them.

$$\begin{aligned} M_I(t_3) = \{&Ca \rightarrow A(lCol);\\ &Na_1 \rightarrow Q(\text{“What is this number?”}, Na_1, numT);\\ &Na_2 \rightarrow Q(\text{“What is this number?”}, Na_2, numT)\} \end{aligned}$$

where Na_1 and Na_2 are the location of the 2 numbers.

From t_3 to t_4, the human operator labels the 2 numbers.

$$\begin{aligned} M_I(t_4) = \{&Ca \rightarrow A(lCol); Na_1 \rightarrow A(num(V_1));\\ &Na_2 \rightarrow A(num(V_2))\} \end{aligned}$$

where V_1 and V_2 are the values of the 2 numbers.

From t_4 to t_5, the page analyzer reprocesses the whole page and finishes the analysis using the answers. It produces a consistent structured interpretation.

4) Implementation Advantages: This implementation of a directed interaction allows to clearly separate the document-related semantic (which part of the model is impacted by an error, which parts are independent) from the iterative error correction based on the visual memory. Apart from enabling another implementation (like synchronous answering), it also permits to make use of human knowledge at various levels in the analysis: field (lexical), presentation and structure (syntactical), and function (semantical).

E. Application to Documents from the 18th Century

The implementation of the iterative analysis we presented is one of the pillars enabling the assisted transcription of handwritten words from sales registers of the 18$^{\text{th}}$ century described in [5]. In the tests we conduced, 70 documents (1206 words) were processed, and extracted words were grouped by visual similarity in clusters, which were manually annotated if necessary. The iterative analysis permitted to reintegrate and validate external information (extracted using collection context) within a unique page model. This global approach diminished the human labeling of words from 21% (manual labeling of all suspicious elements) to 15% (28% gain) for an overall recognition rate of 80%.

V. Conclusion

We presented a solution to the problem of efficiency for the necessary interaction during the analysis of document collections. We showed that: asynchronous exchanges avoids the user or the system to wait (*viability*); keeping a unique page model, separated from temporal issues, inside the page analyzer facilitates its use and enables an homogeneous integration of human information (*simplicity*); and iterative analysis permits reprocessing and other durable modifications of the analysis (*durability*).

We proposed an *iterative approach* for the analysis of document pages, using a *visual memory* to permit an asynchronous information exchange between analysis components, which enables two interaction models: *directed* and *spontaneous*. We implemented and tested the *directed* model on a transcription task in historical documents presented in [5], where the iterative approach permitted to make use, during the page analysis, of information produced using collection-level information. This implementation is based on error recovery principles, very similar to exceptions in their usage, and may be quite easily adapted to turn existing isolated page analyzers into interactive ones.

Acknowledgment

This work has been done in cooperation with the *Archives départementales des Yvelines* in France, with the support of the *Conseil Général des Yvelines*.

References

[1] G. Nagy and S. Veeramachaneni, “Adaptive and interactive approaches to document analysis,” in *Machine Learning in Document Analysis and Recognition*, ser. Studies in Computational Intelligence, S. Marinai and H. Fujisawa, Eds. Springer, 2008, vol. 90, pp. 221–257.

[2] S. Yacoub, V. Saxena, and S. Sami, “PerfectDoc: a ground truthing environment for complex documents,” *Proc. of ICDAR*, vol. 1, pp. 452–456, 2005.

[3] E. Clavier, G. Masini, M. Delalandre, M. Rigamonti, K. Tombre, and J. Gardes, “DocMining: A cooperative platform for heterogeneous document interpretation according to user-defined scenarios,” in *Graphics Recognition*, Lladós and Kwon, Eds. Springer, 2004, vol. 3088 of LNCS.

[4] B. Klein, A. Dengel, and A. Fordan, “smartFIX: An adaptive system for document analysis and understanding,” in *Reading and Learning*, Dengel, Junker, and Weisbecker, Eds. Springer, 2004, vol. 2956 of LNCS.

[5] L. Guichard, J. Chazalon, and B. Coüasnon, “Exploiting Collection Level for Improving Assisted Handwritten Words Transcription of Historical Documents,” in *Proc. of ICDAR*, 2011.

[6] F. Bapst, A. Zramdini, and R. Ingold, “A scenario model advocating user-driven adaptive document recognition systems,” *Proc. of ICDAR*, p. 745, 1997.

[7] B. Coüasnon, “Dealing with noise in DMOS, a generic method for structured document recognition: An example on a complete grammar,” in *Graphics Recognition*, Lladós and Kwon, Eds. Springer, 2004, vol. 3088 of LNCS.